Password Authentication with Secured Login Interface at Application Layer

Vivek Kumar Agrawal †, R. K. Bharti††

Department of CSE, B.T. Kumaon Institute of Technology, India

Summary
In this paper we present a innovative solution to the old problem of password security at application layer or input level. In our solution, each of the various lower case & upper case characters, special characters, digits from which a password could be comprised is encoded with a random single digit integer number and presented to the user through an Login input interface module. A valid user entered his password in the form of the sequence of code or numeric value from the login input interface module that describe his password code in place of his actual password characters. This approach does not require the input code to be hidden from anyone (shoulder surfing) or converted to placeholder characters for security reasons. Our solution engine every time regenerates new numeric value when user enters password for each character each time the carriage return key is struck so our approach is key logger attack protected, producing a toughened password that is convincingly more secure than an old and widely used password scheme is conventional password scheme. In which the system first authenticates the username and password at the login time from the user database and on the basis of authentication of the user, allows the user to access the system. However, such scheme is vulnerable to attacks like Shoulder Surfing, Key loggers, Phishing Attacks and Login Spoofing against both online and offline attackers. our approach is feasible in practice, ease of use, better security and performance.

Key words:
Randomly generated Alphanumeric value, Password authentication, password matching Algorithm.

1. Introduction

A password is a secret word or phrase (combination of alphabet, numbers, special character) that allows the user to access the system resources. The password assists in ensuring that unauthorized user don't access the resources. In the many password mechanism the password length are usually small or short, it makes easier to spy and memorize the passwords via monitoring of keystrokes or through eavesdropping[8]. In an organization or company every user has unique password to use system resources and many employee presence around him. Personal identification number (PIN) is always used by banks to allow their customers access to their online banking facility. Often banks provides to user small length PINS for the customers' convenience that is easy to be remembered by the user but this advantage is create a problem that short length password easier for attacker to memorize short keystroke.

In our solution try to make a try to make a spy-resistant password entry module that looks as a Keyboard or virtual keyboard to improve more security on publicly observable [4, 7]. Our approach give a liberty to user to Passwords secures, is hard to guess and changed frequently. In our solution user enters a randomly generated single integer value in place of characters so it makes much difficult to crack through brute force attack or dictionary attacks because brute force attack attempt to discover/guess the correct password by trying every possible key combinations of letters, numbers and symbols that will unlock the encryption until you find the right one [1]. It could take few minutes or may be years to discover it that is depend on the password's length and complexity; there could be trillions of possible combinations[3]. For example a user enters a password of 10 characters and all characters are lower case letters then to crack this password using the brute force attack it requires (26)10 possible combinations which is equal to 141167095653376. If a single computer takes 1000 passwords to check in one second then total time will be 141167095653376 / 1000 = 141167095653.376 seconds which is equal to 39213082.125938 hours. It means brute force attack is effective only for smaller passwords. Dictionary attacks are very similar to Brute-force attack but little bit faster. These attacks are called hybrid brute-force attacks; it doesn't make combination of input characters but uses dictionary words. These attacks fail breaking lengthy and stronger passwords. Many users mostly choose a passwords related to the cell phone number, famous actors names, date of Birth and birth place, names of pets, familiar places etc [2]. These passwords can be guessed by dictionary attack. Now users may choose only one password for all systems in the case of multiple systems.

In this paper, we are giving an approach or authentication system module algorithm to solve the password security problem at Input Level.

Manuscript received January 5, 2015
Manuscript revised January 20, 2015

2. System Architecture and Algorithm

This is the architecture of our approach. We

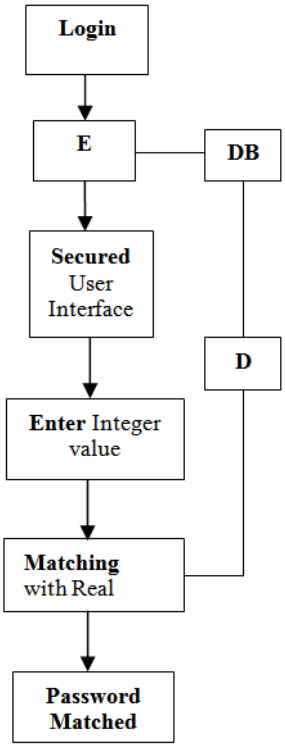

Figure 1: Secured password authentication module Architecture

3. Login Interface Module and Algorithm

Visual Studio 10 used for the implementation of the algorithm. From the algorithm, a Login interface module was generated figure:3. The module shows the password characters with their corresponding single integer value (in black box). User enter the numeric input value according to his password in password text box area, if a user take too more time to enter password then login interface will be refresh and the randomly generated integer values corresponding to character, digits, special characters wille be changed or user can do this itself on clicking "Regenerate random number set " button.

3.1 In this, first user registered name, user id and password that is stored in database in encrypted form. The database DB, contains all the passwords of legitimate users. The database is designed to accept 255 characters length passwords but coder must restrict size of users' passwords to a reasonable length, for example 14 characters, for better security.

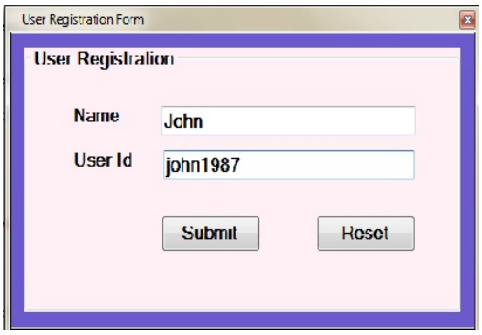

Fig 2.1: User registration form

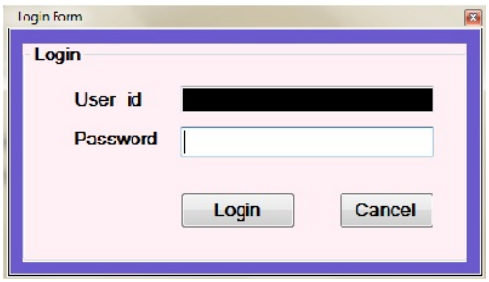

Fig 2.2: User registration form

3.2 Then user login with user id and password using secure login user interface. In which all printable ASCII characters available, which may consist of lower and upper case alphabets (A-Z, a-z), numeric digits (0-9), and special characters (+ - _ ^ , # % etc).

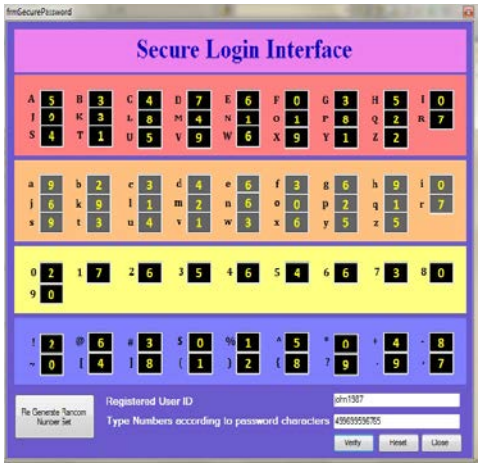

Fig 3: Secure Login Interface

3.3 User enters a single integer value corresponding to each characters according to his password. Every time a new single integer value (0 to 9) generated corresponding to all characters, digits, special character whenever user logging that .

```
private int RandomNumber(int min, int max) {
        return random.Next(min, max);
        }
```

3.4 User enter password in the form of randomly generated single integer value correspond to valid password characters after studying the secure login user interface,
If user password length = = original password is false then a message "incorrect password, must be same" shown.
If condition True means the integer value that is entered corresponding to password is moved to matching with original password. For this we use a jagged array[ ] in which all user input integer values stored and we create another array name occurance[ ] for to store each characters separately accordingly to their correspond single integer value zero to nine that is randomly generated it means it may be assign to more than one characters.

```
if (txtPassword.Text.Length == General. userpass.Length
&&     txtPassword.Text.Trim().Length>0) {
            m=10;
       jaggedArr = new string[m][];
          for (k = 0; k < m; k++) {
         int s = occurance[k];
           jaggedArr[k] = new string[s];
       } fillJaggedArray();
         MatchPassword();
         } else
    {   MessageBox.Show
       ("Incorrect Password Length Must be  Same");
    } }
```

Single input integer value correspond to each valid password characters

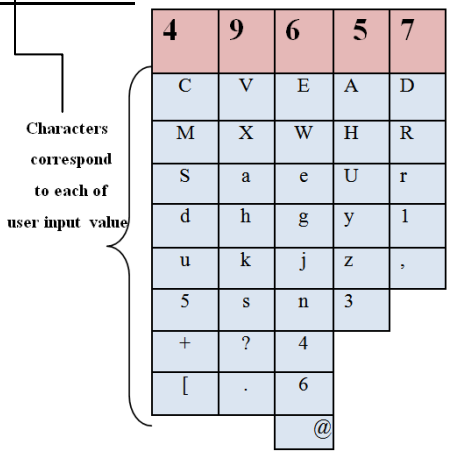

Fig 4: Example of Jagged array

This an example of How our algorithm create a table separately for every integer input value corresponding to their character.

```
for (int j1 = 0; j1 < 10; j1++) {
        occurance[j1] = 0;
} while (i < 80) {
   num = RandomNumber(00, 10).ToString("D1");
   numarr[i] = num;
   i++;
   ch =Convert.ToInt32(num);
   switch (ch) {
      case 0:
         occurance[0]++;
         break;
      case 1:
         occurance[1]++;
         break;
      case 2:
         occurance[2]++;
         break;
      case 3:
         occurance[3]++;
         break;
      case 4:
         occurance[4]++;
         break;
      case 5:
         occurance[5]++;
         break;
      case 6:
         occurance[6]++;
         break;
      case 7:
         occurance[7]++;
         break;
      case 8:
         occurance[8]++;
         break;
      case 9:
         occurance[9]++;
         break;
     }          }
private void fillJaggedArray(){
 for (int x = 0; x < 10; x++) {
   int z = 0;
    for (int y = 65; y <= 90; y++) {
     string txtCap = "txt" + (y).ToString();
     Control[] txtC = this.Controls.Find(txtCap, true);
    if (txtC != null && txtC.Length > 0) {
      if (txtC[0].Text.CompareTo(x.ToString()) == 0) {
          jaggedArr[x][z] =Convert.ToChar(y).ToString();
              z++;
           }}}
```

3.5 Our algorithm match the user input value to the actual password one by one character. Suppose our password is "Chanakya@123" and user input value corresponding there is 499699596765 so for this our algorithm create a table for every user input value fig.4. First we match user input value 4 with actual password first character 'C' if this match then go to next otherwise move out and a message is shown " Password Not Matched " and if first character is matched then this process move next until we find actual password.

```
private void MatchPassword() {
    bool flag=false;
    string test = "";
   for(int w=0;w<txtPassword.Text.Length;w++) {
      int d;
      flag = false;
 d=Convert.ToInt32(txtPassword.Text.Substring(w,1));
 for(int v=0;v<jaggedArr[d].Length;v++) {
    if (General.userpass.Substring(w, 1). CompareTo
 (jaggedArr[d][v]) == 0) {
       test = test + jaggedArr[d][v];
            flag = true;
            break;
           }            }          }
if (flag == true) {
    MessageBox.Show("Password has been Matched");
    MessageBox.Show("Your Password was : " + test);
 }
  else {
       MessageBox.Show("Password Not Matched");
    }       }
```

4. Conclusion

In this paper we have been given our approach Randomly generated single integer input digits corresponding to password characters on login interface module makes it impossible for for attackers to hack or electronically eavesdrop, shoulder surfing, brute force attack on user password at input level(at application layer). It will improve the security and integrity of the password systems.

References

[1] Fujita, K. and Y. Hirakawa, 2008, "A study of password authentication method against observing attacks",6th International Symposium on Intelligent Systems and Informatics, SISY 2008.

[2] Kessler, Gary C., 2002. Passwords - Strengths and Weaknesses. Jan-1996.URL: http://www.garykessler.net/library/password.html

[3] G. Sowmya, D. Jamuna, M.Venkata Krishna Reddy, "Blocking of Brute Force Attack" International Journal of Computer Applications & Information Technology, Vol. I, Issue II, September 2012.

[4] Desney S. Tan, Pedram Keyani, Mary Czerwinski ."Spy-Resistant Keyboard: Towards More Secure Password Entry on Publicly Observable Touch Screens".

[5]

[6] Schneider, B., "Applied Cryptography Second Edition: protocols, algorithms, and source code in C", John Wiley & Sons, Inc., 1996.

[7] Mathias Kolsch, Matthew Turk. "A Survey of Virtual Keyboards", Dept. of Computer Science, University of California at Santa Barbara, CA.

[8] [Mark, 2005] Mark S., (2005), "Information Security, Principles and Practice", Wiley Interscience.

[9] I. Scott MacKenzie, "KSPC as a Characteristic of Text Entry Techniques", Dept. of Computer Science, York University Toronto, Ontario, Canada M3J 1P3.

Vivek Kumar Agrawal received the B.Tech. degree in Computer science & Engineering from AAIDU in 2008 and pursuing M.Tech from department of CSE, B.T.Kumaon Institute of Technology, India.

R.K.Bharti received PhD in Data Compression department of CS. He has published many research paper in Cryptography & network Security and Data compression area. He is working in department of CSE, B.T.Kumaon Institute of Technology, India as an Assistant Professor since 10 years.